

## LA-UR-13-20948

Approved for public release; distribution is unlimited.

Title: PARTISN on Advanced/Heterogeneous Processing Systems

Author(s): Baker, Randal S.

Intended for: Public dissemination.



### Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

# Abstract

---

- This presentation describes both the importance and the difficulty of parallelizing deterministic neutron transport calculations using the LANL code PARTISN. A brief history of the standard "KBA" parallelization method used for deterministic SN transport is presented, then the modifications necessary to implement this on Roadrunner are covered. The use of the Roadrunner algorithm with OpenMP is then described, along with a brief mention of an alternative thermal radiation transport algorithm for GPUs. Finally, ongoing work with PARTISN on the MIC many-core architecture is noted.

# PARTISN on Advanced/Heterogeneous Processing Systems

Randy Baker

*Computational Physics Group (CCS-2)*

*Los Alamos National Laboratory*

# Deterministic, time-dependent neutron transport forms a large part of the overall ASC workload

---

- Neutron transport is inherently seven-dimensional (three in space, two in angle, one in energy, and one in time)
- Deterministic neutron transport has always stressed every aspect of computational performance since the 1950's
- **PARTISN, 1D/2D/3D static or time-dependent transport package, origins late 50's**
- Large number of floating point operations (vectorization/cache size)
- ~1 FLOP/load (memory bandwidth)
- Frequent small messages (MPI latency)
- But not that small... (MPI bandwidth)
- Large array/data sizes (memory storage)
- Large restart files (disk performance)

# It's better to be a dog in a peaceful time than be a man in a chaotic period.....

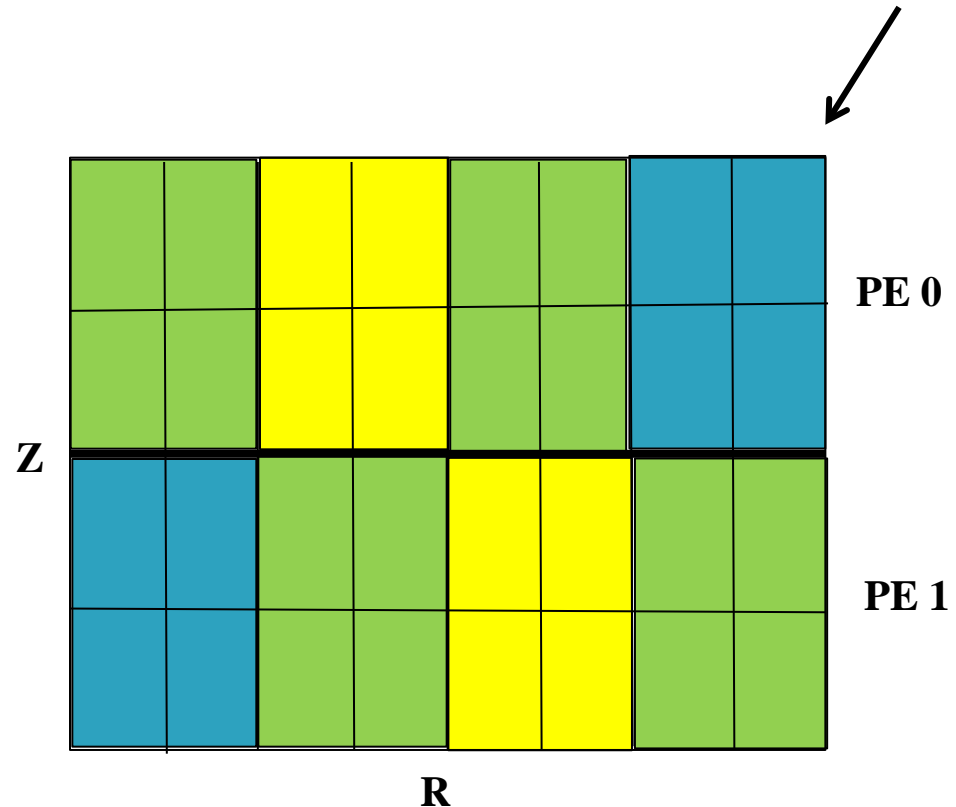
---

- Early 1990's, transition from "traditional" Parallel Vector Processors (e.g., Cray Y-MP) to today's MPI clusters
- Data parallel vs. message passing
- CM-Fortran vs. HDF vs. MPICH vs. MPI vs. ....
- Similar uncertainty about viability of "traditional" algorithms on new platforms
- **Transport sweeps parallelized through use of "KBA"**
- Days of "MPI everywhere" clusters leading HPC are already ending at national labs
- "Bleeding edge" platforms are heterogeneous, i.e., traditional RISC processor plus.....  
**Cell/GPU/FPGA/many-core**
- MPI still used, but in combination with threads/DaCS/???
- F95/C++ and CUDA/OpenCL/???

Are "KBA" transport sweeps still viable?

# “KBA” sweeps are well-suited for 3-D geometries, but inherently problematic in 2-D geometries

- Wavefront technique requires a 1-D spatial domain decomposition in 2-D R-Z geometries
- Idle processors as sweep starts at one end of the mesh and finishes at the other
- Standard “KBA” sweep increases “busy” fraction of processors by “stacking” angles on top of R-axis spatial cells
- This technique has proven to provide effective parallelization out to hundreds of traditional processors for 2-D R-Z calculations



Can we scale 2-D KBA sweeps on heterogeneous (many core) architectures?

# The increase in “compute cores” on the Cell architecture challenges 2-D “KBA” sweeps

---

- Each Opteron “Host” core is now associated with one PPE on the Cell
- Each PPE has 8 SPEs
- Each SPE is essentially a vector processor with a vector length of two
- Performance gains on the Cell are achieved by using the SPEs in vector mode
- **This effectively increases our “compute core” count by 8X (16X)**
- For a fixed problem size, increasing the number of processors degrades the PCE of “KBA” sweeps
- Previous work at implementing “KBA” sweeps for the Cell judged not effective for 2-D geometries

More “compute cores” means we also need more independent data streams

## Vectorization over angles/polar levels and “threading” over energy groups allows us to maintain acceptable 2D/3D parallel performance

---

- “Stacking” of angles in traditional “KBA” sweeps has now effectively been replaced with “stacking” of energy groups
- Each thread only works on a single energy group at a time, facilitating implementation of group-dependent quadratures
- The spatial chunk size used by the MPI rank to communicate to downstream processors has been reduced
- This increases the communications frequency, but the MPI rank is no longer compute-bound while performing the sweep, so communications latency is not an issue (?)

MPI in space  
“threaded” over energy  
vectorized over angles

## 2-D transport sweeps can now be efficiently performed on Roadrunner using the “Cell KBA” algorithm

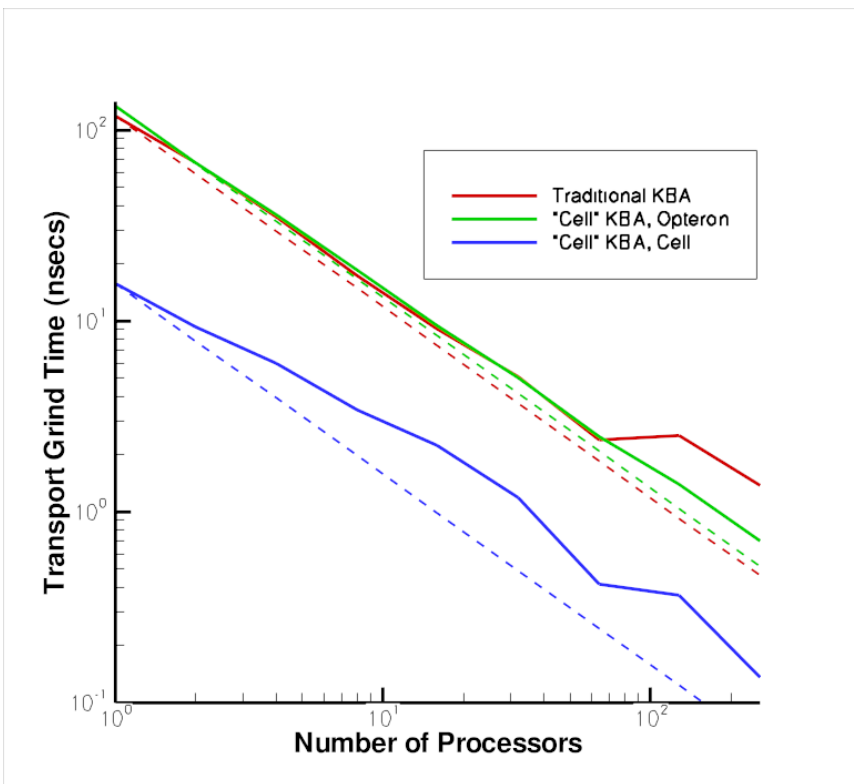
- 180x360 spatial mesh,  $S_{16}$  square C-L(64 angles/octant), spatial “chunk” size of 90 on Opteron, 8 on Cell
- 70 energy groups
- PCE for traditional “KBA” is not a function of number of energy groups
- PCE for “Cell KBA” is now a function of energy groups, but no longer a function of  $S_N$  order

$P$	PCE	Cell PCE
1	1.000	0.951
2	0.996	0.949
4	0.988	0.944
8	0.973	0.935
16	0.945	0.918
32	0.892	0.885
64	0.803	0.825
128	0.668	0.728
256	0.501	0.589
512	XXXXX	XXXXX

“Cell” KBA sweep using 2,048 “compute cores”

# The “Cell KBA” sweep is even more effective for 3-D geometries on Roadrunner

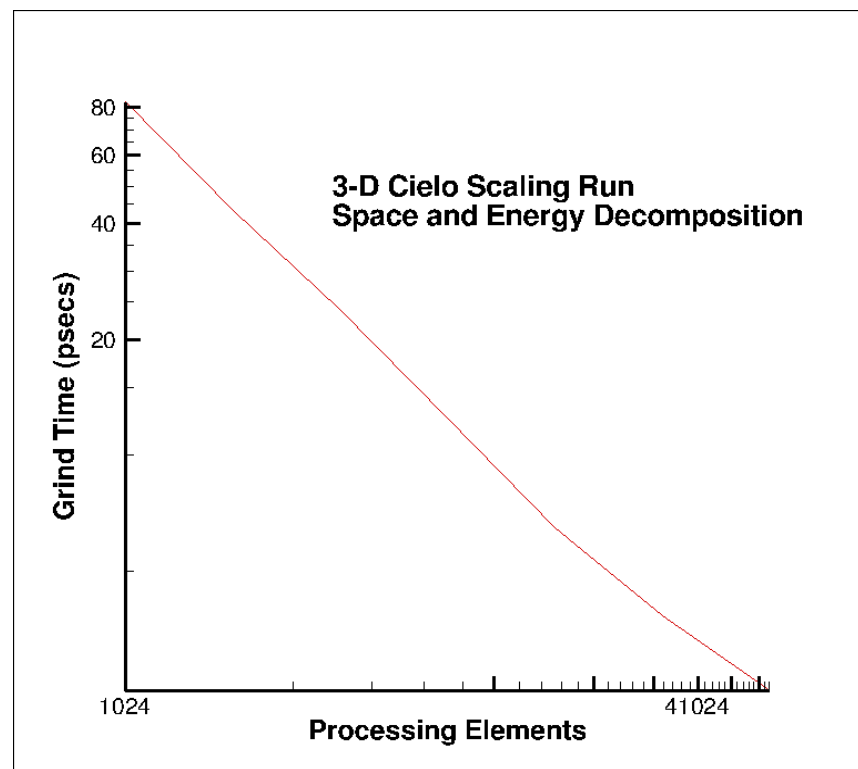
- Weak scaling study (~13,700 cells per processor)
- Cube of Pu with left/bottom/front reflecting boundary conditions, constant  $dx$ ,  $k_{eff}$
- $S_{16}$  triangular quadrature (36 angles/octant),  $P_2$  scattering, 30 energy groups), spatial “chunk” size of 10 on Opteron, 2 on Cell
- Transport Grind Time is time for transport operator per phase space cell



Cell speedup from ~4X to over 8X for 3-D geometries

## Even with 3-D geometries, “KBA” sweeps have already shown need for an “MPI+X” programming model on other platforms

- Models show “MPI everywhere” should be efficient for 3-D “KBA” sweeps out to ~100,000 cores
- Large amounts of nuclear data replicated, already insufficient memory on one node of Luna for some 1-D calculations
- Use of MPI+OpenMP (one node, one MPI rank, 16 threads) allowed 1-D calculations to run
- Large 3-D calculation on Cielo, still “MPI everywhere”, but needed spatial **AND** energy decomposition



## Implementation of the Cell algorithm within PARTISN was a team effort

---

- No significant differences in performance or code text size between C and Fortran, used (mostly) Fortran for PPE and SPE functions for maximum reuse of existing Fortran
- Initial development begun Spring 2009 with separate Cell-only CVS repository, transferred very quickly into branch of PARTISN CVS repository
- Cell branch merged onto PARTISN trunk January 2011
- PARTISN ~117,000 Source Lines of Code (SLOC)
- Currently ~3,900 SLOC for PPE/SPE-specific functions
- Additional ~2,300 SLOC in “standard” PARTISN to use these Cell functions

Hardware is (almost) “cheap”, software is expensive.....

## **SPE Heterogeneous programming leads one to question their career choices and increase their alcohol consumption.....**

---

- Asynchronous memory puts/gets, controlled by the SPE, used to transfer data from/to main memory
- Checking for arrival of data on SPE prior to use not sufficient, also had to check that it had been stored in main memory before request (2X slowdown)
- Buffering scheme used to overlap data transfers and computations
- Number of buffers arbitrary, small-scale testing shows ~10% increase in performance over one buffer with two buffers, little additional with three, and performance degradations thereafter
- Default setting is two buffers, use of more not recommended due to memory limitation of only 256 Kbytes per SPE for OS/Code text/code data!
- Code overlays used to reduce SPE memory usage

**STOP!**

**But maybe not for multi-physics???**

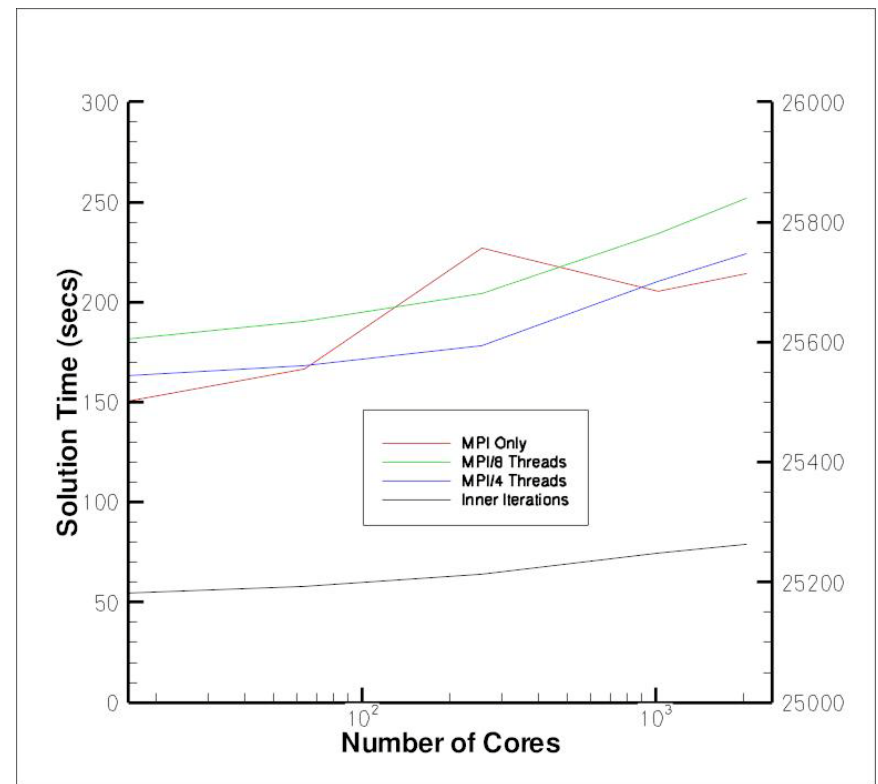
# Threading (OpenMP) is not yet widely used (supported) by ASC at LANL

---

- LANL “how to” documentation almost non-existent, especially on how to place/bind MPI ranks (and their threads) on NUMA nodes
- PGI compiler not working in MPI+OpenMP mode due to libnuma.a issues
- Intel compiler only supports binding of threads to cores (performance!) on Intel hardware, and even there still had to be disabled for Luna
- Floor version of OpenMPI not yet truly “thread safe”

## Transitioning PARTISN from Cells/SPEs to MPI+OpenMP was straightforward and the code is now in production use

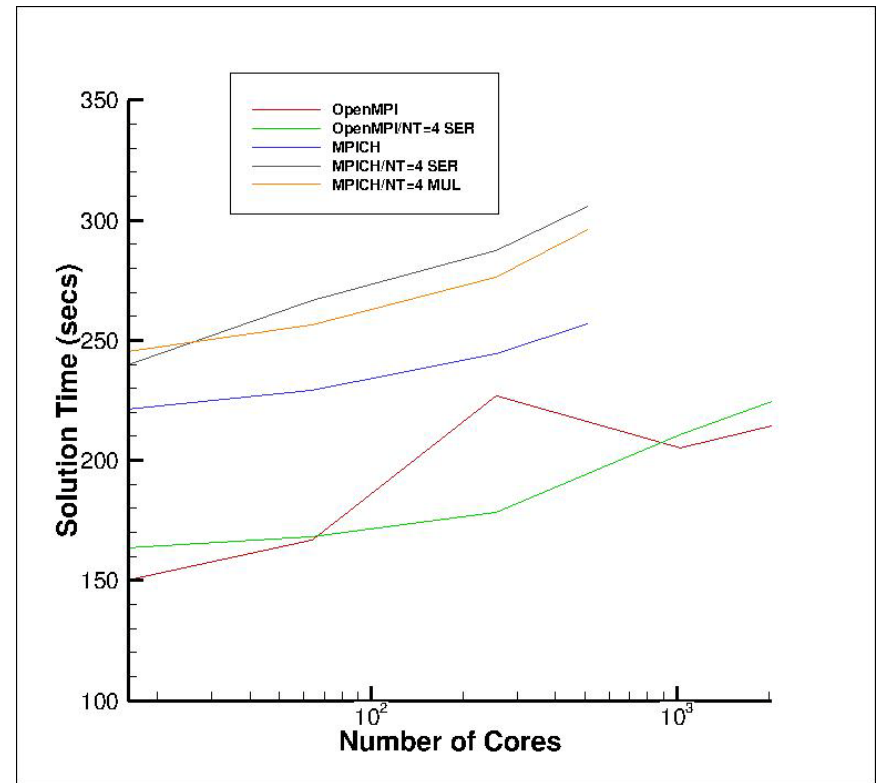
- ~1 week effort, “shadow” Opteron code converted to use threading (OpenMP) over energy, MPI over space and vectorization in angle retained
- Significant advantages over “MPI everywhere” for 1-D calculations
- OpenMP ~10-20% faster, reduced memory usage, for 1-D calculations
- Still problematic for 2-D/3-D calculations



**2-D scaling study on Luna**

# “KBA” sweeps combined with threading over energy requires a truly thread-safe MPI library for performance

- **MPI\_THREAD\_SINGLE** - Only one thread will execute
- **MPI\_THREAD\_FUNNELED** - The process may be multi-threaded, but only the main thread will make MPI calls
- **MPI\_THREAD\_SERIALIZED** - The process may be multi-threaded, and multiple threads may make MPI calls, but only one at a time
- **MPI\_THREAD\_MULTIPLE** - Multiple threads may call MPI, with no restrictions



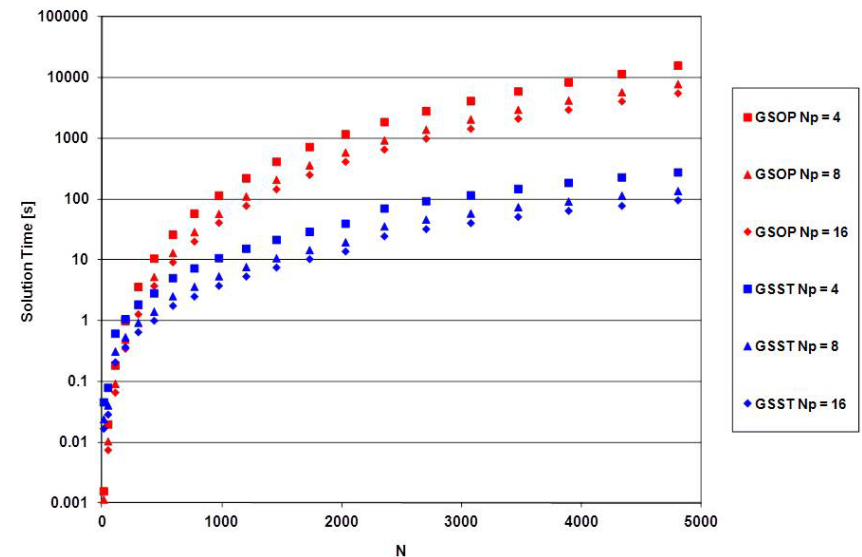
User locks!

Library locks?

2-D scaling study on Cielito

# PARTISN (“KBA” sweeps) on GPUs is a problematic proposition

- Insufficient independent data streams in 2-D, and probably even in 3-D, to keep GPU busy
- Combination of data and task parallelism may help, difficult to program/maintain?
- Abandoning “KBA” sweeps puts deterministic neutron transport into uncharted territory, substantial verification/validation effort
- More “GPU-friendly” algorithms showing promise for deterministic thermal radiation transport



**Capsaicin scaling study on  
Moonlight, with and without  
GPUs**

# Implementation of PARTISN on MIC's (Knights Corner) is currently underway

---

- Heterogeneous mode, not native (two months wasted.....)
- Focus is on optimizing single core performance on the MIC
- Built with OpenMP on host and MIC, single thread testing only (so far)
- No MPI (only two Knights Corner nodes at LANL)
- Targeted at “production” implementation based on OpenMP algorithm <- Roadrunner algorithm

## Initial performance (after two weeks) on MIC shows much work yet to done on improving vectorization by both Intel and PARTISN

- MIC, 60+ cores, vector length (64-bit) of eight
  - Host, Sandy Bridge (?), vector length of two
1. Auto-vectorization flags, compiler directives, **memory alignment**
  2. Data reorganization within PARTISN
  3. MIC vector intrinsics

	Host	MIC
Initial	79.2 secs	655.7 secs
-no-vec	1.25X	1.65X
Step 1(b)	71.9 secs	515.7 secs
-no-vec	1.38X	1.90X

## Lessons learned (AKA “survival strategies”) in programming for advanced architectures

---

- Minimize data movement, whether to/from main memory or to/from host/accelerator
- Maximize the number of independent data streams within your algorithms (and/or change your algorithms!)
- Vectorize, and then vectorize some more

*be prepared to throw away your code every few years.....*

*play well with others.....*